

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-351679  
(43)Date of publication of application : 06.12.2002

(51)Int.Cl. G06F 9/46  
G06F 1/00

(21)Application number : 2001-158949  
(22)Date of filing : 28.05.2001

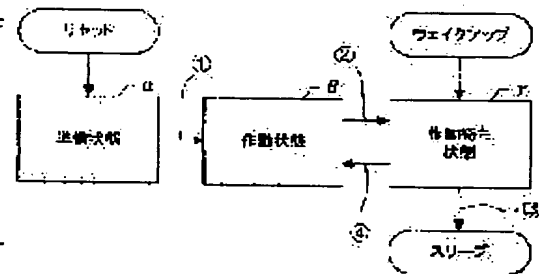
(71)Applicant : DENSO CORP  
(72)Inventor : NITTA SHUICHI  
MATSUDA KEISUKE  
EGAWA KUNITAKA

## (54) PROGRAM AND ELECTRONIC CONTROLLER

### (57)Abstract:

**PROBLEM TO BE SOLVED:** To provide a program, etc., capable of reducing resource consumption of an application by reducing costs for design and mounting of the application.

**SOLUTION:** Execution results are stored in the case of executing the respective applications and when the current state is in a preparation state  $\blacksquare$ , the state is transited to an activated state  $\blacksquare$  when preparation of all the execution results of APL is completed in a control state judgment processing to be executed after execution of all the APLs. In the activated state  $\blacksquare$ , the state is transited to an activation standby state  $\blacksquare$  when control of the execution results from all the APLs is completed (sign 2). In the activation standby state  $\blacksquare$ , the state is transited to the activated state  $\blacksquare$  when the control of execution results of any of the APL is started (sign 4). On the other hand, when no start of control exists in the execution results of the respective APLs, the state is transited to a sleep to stop operations of the ECU (sign 3). In the control state judgment processing, previously registered routines in the respective APLs are successively called executed in the case of transition to the respective states.



5

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2002-351679

(P2002-351679A)

(43)公開日 平成14年12月6日(2002.12.6)

(51)Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード <sup>*</sup> (参考)
G 0 6 F 9/46	3 4 0	G 0 6 F 9/46	3 4 0 B 5 B 0 9 8
1/00	3 7 0	1/00	3 7 0 D

審査請求 未請求 請求項の数7 O L (全 6 頁)

(21)出願番号 特願2001-158949(P2001-158949)

(22)出願日 平成13年5月28日(2001.5.28)

(71)出願人 000004260

株式会社デンソー

愛知県刈谷市昭和町1丁目1番地

(72)発明者 新田 修一

愛知県刈谷市昭和町1丁目1番地 株式会  
社デンソー内

(72)発明者 松田 啓資

愛知県刈谷市昭和町1丁目1番地 株式会  
社デンソー内

(74)代理人 100082500

弁理士 足立 勉

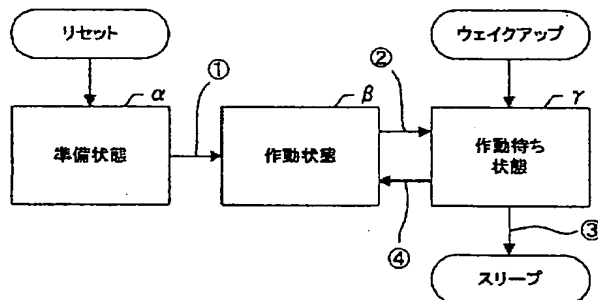
最終頁に続く

(54)【発明の名称】 プログラム、電子制御装置

(57)【要約】

【課題】アプリケーションの設計及び実装のコストを軽減しアプリケーションのリソース消費を減らすことのできるプログラム等を提供する。

【解決手段】各アプリケーション実行時に実行結果を記憶しておき、すべてのAPLの実行後に実行する制御状態判定処理で、現在の状態が準備状態 $\alpha$ にある場合には、すべてのAPLの実行結果が準備完了の場合、作動状態 $\beta$ へ状態を遷移させる。作動状態 $\beta$ では、すべてのAPLから実行結果が制御完了の場合、作動待ち状態 $\gamma$ へ状態を遷移させる(記号②)。作動待ち状態 $\gamma$ では、いずれかのAPLの実行結果が制御開始の場合、作動状態 $\beta$ へ状態を遷移させる(記号④)。一方、各APLの実行結果に制御開始がない場合、ECUの動作を停止するスリープへ遷移させる(記号③)。制御状態判定処理では各状態遷移時に各APL内の予め登録されたルーチンを順次呼び出して実行する。



## 【特許請求の範囲】

【請求項 1】 コンピュータに所定の機能を実現させるためのプログラムである複数のアプリケーションと、前記アプリケーションを呼び出す処理を実行させるプログラムであるプラットフォームとを含むプログラムであって、

前記プラットフォームは、前記各アプリケーションの実行結果に基づいて所定の状態を管理する機能を実現させるためのプログラムを含み、

前記アプリケーションには、前記プラットフォームによって管理される前記所定の状態を参照するプログラムを含むものがあり、

前記アプリケーションには、前記所定の状態が変化した場合に実行すべきプログラムである所定のルーチンを含むものがあり、

前記プラットフォームは、前記実行結果に基づいて前記所定の状態を遷移させた際に、前記所定のルーチンを実行させるためのプログラムを含むことを特徴とするプログラム。

【請求項 2】 請求項 1 に記載のプログラムにおいて、前記所定の状態の遷移には、コンピュータをスリープさせるための状態遷移が含まれることを特徴とするプログラム。

【請求項 3】 請求項 1 または 2 に記載のプログラムにおいて、

前記所定の状態の遷移には、準備状態から作動状態への状態遷移、作動状態から作動待ち状態への状態遷移、作動待ち状態から作動状態への状態遷移の少なくともいずれか 1 の状態遷移を含むことを特徴とするプログラム。

【請求項 4】 請求項 1 ～ 3 のいずれかに記載のプログラムにおいて、

前記プラットフォームは、前記アプリケーションの全ての前記実行結果が状態遷移可能である旨を示す場合に、前記所定の状態を遷移させることを特徴するプログラム。

【請求項 5】 請求項 1 ～ 4 のいずれかに記載のプログラムにおいて、

前記プラットフォームは、前記アプリケーションのいずれかの前記実行結果が状態遷移要求を示す場合に、前記所定の状態を遷移させることを特徴するプログラム。

【請求項 6】 請求項 1 ～ 5 のいずれかに記載のプログラムにおいて、

前記プラットフォームは、所定周期毎に実行され、前記アプリケーションの呼び出しを順次行わせ、前記全アプリケーションの実行終了後に前記所定の状態の遷移を行うか否かを決定させることを特徴とするプログラム。

【請求項 7】 請求項 1 ～ 6 のいずれかに記載のプログラムを実行するコンピュータを備えた電子制御装置。

## 【発明の詳細な説明】

【0001】

【発明の属する技術分野】 プログラム等に関する。

【0002】

【従来の技術】 従来よりコンピュータシステムにおいて、メインプログラム（例えば OS）と複数のアプリケーションプログラムとを備えて、メインプログラムによる処理によって各アプリケーションプログラムを呼び出して実行することにより、各アプリケーションプログラムに基づく所定の機能をコンピュータシステムに実現させることが行われている。

10 【0003】 このようなコンピュータシステムでは、各アプリケーションプログラムによる処理において、相互に他のアプリケーションプログラムの状態を参照し、その状態と自己の状態に基づいて処理を実行することがある。例えば、図 2 に示すように、所定時間毎に周期的に実行されるメインプログラムであるプラットフォームから、複数のアプリケーションプログラム（以下 APL とも称する）を順次呼び出して実行する場合に、それぞれの APL は、前回の実行周期における実行状態を記憶しておき、記憶されたそれぞれの APL の実行状態をそれぞれ

20 ぞれの APL で参照して、その状態に基づいてそれぞれの APL で状態を遷移させるか否かを判定する。そして状態が遷移したと判定した場合には、その APL のプログラムの一部である状態遷移時の処理ルーチンを実行する。

【0004】

【発明が解決しようとする課題】 しかしながら、例えば状態としてスリープ・ウェイクアップや通信の開始・停止など、システム全体で 1 の状態を取るもののように各 APL で共通して参照する状態についても、それぞれの APL において相互に各 APL の状態に基づいて状態を遷移させるか否かの状態判定を行っているため、コンピュータシステムのリソースを無駄に消費してしまうといった問題があった。

【0005】 また、各 APL 内で状態遷移が発生した場合に実行したいルーチンがある場合には、状態遷移の発生を捉える必要があるため、以前の状態を記憶するための領域や、この記憶領域に記憶された状態と現在の状態に基づいて状態の変化があったか否かを検出するためのプログラムを各 APL 毎に備える必要がある。そのため

40 APL のリソース（検出プログラムを格納するメモリ、状態を格納するメモリ、プログラムの処理時間を含む。）の消費が APL が増えれば増えるほど大きくなり、本来の APL による機能を実現させるためのリソースの量と比べても、状態の変化を検出させるために必要なリソースの量が無視できないほど多くなる可能性がある。

【0006】 しかも、相互に APL の状態を参照するようにすると、APL を増やすたびに他の APL のプログラムを修正する必要があるため、設計や実装にコストがかかるといった問題がある。そこで本発明は、アプリケ

50

ーションの設計及び実装のコストを軽減し、アプリケーションのリソース消費を減らすことのできるプログラム等を提供することを目的とする。

【0007】

【課題を解決するための手段及び発明の効果】 上述した問題点を解決するためになされた請求項1に記載のプログラムによれば、各アプリケーションの実行結果に基づいてプラットフォームによる処理で状態が管理される。したがって、各アプリケーションによる処理では、このプラットフォームで管理された状態を参照して処理を行うようにするだけでよい。すなわち従来のように、それぞれのアプリケーションにおいて他のアプリケーションの実行結果（状態）を確認し、現在の状態を求める必要がなくなる。

【0008】 よって、アプリケーションの数が増えても、こうした現在の状態を求めるプログラムを格納するために必要なメモリやこのプログラムを実行するための時間が必要となることはない。また、従来アプリケーション毎に設けていた状態遷移を検出するための記憶領域や処理をなくすこともできる。このようにアプリケーションのリソースの消費を減らすことができる。

【0009】 また、従来、相互にアプリケーションの状態を参照するようにすると、アプリケーションを増やすたびに他のアプリケーションのプログラムを修正する必要があるため、設計や実装にコストがかかるといった問題があったが、このような問題もなくなり、システム全体で協調して動作する必要のある状態についての設計や実装を簡便に行うことができ、アプリケーションの管理も容易になる。

【0010】 このようなプラットフォームによる処理で管理すべき状態遷移としては、例えば、請求項2に示すようにコンピュータをスリープさせるための状態遷移がある。コンピュータをスリープさせるか否かは、各アプリケーションにおいて協調して決定する必要があり、状態遷移が可能であるかは各アプリケーションの実行結果に依存する。しかもスリープさせるために状態を遷移させた場合には、各アプリケーションにおいてスリープに備えるための処理（所定のルーチンの実行）を行う必要がある場合が多いからである。

【0011】 こうしたアプリケーション間で協調して制御すべき状態としては、例えば請求項3に示すような状態遷移などが挙げられる。そして、このような状態遷移は、例えば仕様に基づいて所定のアプリケーションの実行結果が所定の内容となった際に行うこともできる。また請求項4に示すようにすべてのアプリケーションによる実行結果が状態遷移可能を示す場合にその所定の状態を遷移させるようにすることもできる。例えば、請求項3のプログラムを前提とする場合、全てのアプリケーションによる実行結果が準備完了になった場合に準備状態から作動状態へ遷移させたり、全てのアプリケーション

による実行結果が制御終了を示す場合に作動状態から作動待ち状態へ遷移させたりすることができる。

【0012】 一方、請求項5に示すようにいずれかのアプリケーションによる実行結果として状態遷移要求があった場合に状態を遷移させるようにすることもできる。例えば、請求項3のプログラムを前提とする場合、いずれかのアプリケーションによる実行結果が制御開始になった場合に作動待ち状態から作動状態へ遷移させることができる。

10 【0013】 また、プラットフォームの例としては、請求項6に示すように、所定周期毎に実行されアプリケーションプログラムを順次呼び出して実行するためのプログラムを含み、全アプリケーションの実行終了後に前記所定の状態の遷移を行うか否かを決定させるようにすることもできる。

20 【0014】 なお、請求項7に示すように、請求項1～6のいずれかに記載の機能をコンピュータシステムに実現させる場合、例えば、コンピュータ側で起動するプログラムとして備えることができる。このようなプログラムの場合、例えば、フロッピー（登録商標）ディスク、光磁気ディスク、CD-ROM、ハードディスク、ROM、RAM等のコンピュータ読み取り可能な記録媒体に記録し、必要に応じてコンピュータにロードして起動することにより用いることができ、また、ネットワークを介してロードして起動することにより用いることもできる。

【0015】

30 【発明の実施の形態】 以下、本発明が適用された実施例について図面を用いて説明する。なお、本発明の実施の形態は、下記の実施例に何ら限定されることなく、本発明の技術的範囲に属する限り種々の形態を採りうることは言うまでもない。

【0016】 実施例の電子制御装置（以下ECUとも称する）は、CPU、ROM、RAM、I/O等をもつマイコンを備えており、CPUがROM及びRAMに記憶されたプログラム及びデータに基づいて、I/Oを制御し、I/O接続された周辺装置を制御する。

40 【0017】 また、このマイコンはスリープ機能を備えており、CPUのスリープ命令の実行により、CPUのクロックが停止され、消費電力を低減するスリープ状態となる。またI/Oからの割込み等に応じてスリープ状態からウェイクアップする機能を備える。I/Oからの割込みとしては、タイマ割込みや、I/Oに接続された通信インターフェースやスイッチ等からの割込みなどがある。

50 【0018】 図1は、ECUの動作概要を示すフローチャートである。ECUは、例えば電源が供給された際や、別のECUとのデータ通信により、リセット指示又はウェイクアップ要因を判断すると動作を開始する。リセット指示があった場合、リセット時のシステム設定を

行い（S100）、その後、S120へ移行する。一方、ウェイクアップ要因を判断した場合、ウェイクアップ時のシステム設定を行い（S110）、その後、S120へ移行する。S120では、各APLによる制御が行われる。ここでは、複数のAPLの状態が判定され、所定条件成立時にスリープ時のシステム設定を行い（S130）、その後、スリープする。

【0019】ここでS120における各APLによる制御を示すのが図2の説明図である。ここには、プラットフォーム内のスケジューラによって、APL1～nの複数のAPLが順次起動される様子が示されている。このとき、プラットフォームにおける制御状態判定処理で、一連のAPL1～nのAPLの状態をまとめて管理し、各APLからの実行結果に基づいて状態遷移を行う。図2の一連の処理は所定の時間間隔（例えば5ms）毎に実行が開始され、図2の一連の処理はこの所定の時間より短い時間に完了するように構成されている。

【0020】各APLには、所定の状態遷移があった場合に実行すべきルーチンが含まれ、このルーチンへの関数ポインタがプラットフォームに登録されている。また各APLはこの制御状態判定処理によって管理される状態を参照して処理を行う。制御状態判定処理による状態遷移の様子を図3に示す。制御状態判定処理では、APLの状態として3つの状態、すなわち準備状態α、作動状態β、そして、作動待ち状態γを定義しており、図2に示したプラットフォームのスケジューラによる一連のAPLの実行が全て終了する毎に、各APLからの実行結果に基づき、必要に応じて状態を遷移させる。なお、各APLから制御状態判定への実行状態の通知は、各APLの処理で実行状態を所定のメモリ領域に書き込み、制御状態判定処理でそのメモリ領域を参照することで行う。

【0021】制御状態判定処理では、リセット指示によって動作を開始している場合、初期状態として準備状態αとし、一方、ウェイクアップ要因を判断して動作を開始している場合、初期状態として作動待ち状態γとする。そして、準備状態αにある場合、すべてのAPLから実行結果として準備完了の通知があると、作動状態βへ状態を遷移させる（記号①）。この遷移を行った場合には、各APL内の準備状態αから作動状態βへの遷移時に実行すべきルーチンをプラットフォームに予め登録された関数ポインタに基づいて順次呼び出して実行する。

【0022】また、作動状態βにある場合、すべてのAPLから実行結果として制御完了の通知があると、作動待ち状態γへ状態を遷移させる（記号②）。この遷移を行った場合には、各APL内の作動状態βから作動待ち状態γへの遷移時に実行すべきルーチンをプラットフォームに予め登録された関数ポインタに基づいて順次呼び出して実行する。

【0023】また、作動待ち状態γにある場合、いずれかのAPLから実行結果として制御開始の通知があれば作動状態βへ状態を遷移させる（記号④）。この遷移を行った場合には、各APL内の作動待ち状態γから作動状態βへの遷移時に実行すべきルーチンをプラットフォームに予め登録された関数ポインタに基づいて順次呼び出して実行する。一方、各APLからの制御開始の通知がなければ、ECUの動作を停止するスリープへ遷移させる（記号③）。

【0024】例えば、ECUが自動車のボデー系のECUである場合、各APLは、ワイヤレス処理、ドアロック処理等を行うためのボデー系装置の制御プログラムである。そして、ECUへのバッテリーからの電源供給後（リセット後）、それぞれのAPLによって制御対象の装置や使用するメモリの初期化、EEPROM等の外部不揮発メモリに保存されている情報の取得等の初期化処理を行う。各々のAPLは自己の初期化処理が終了すると準備完了の通知を行う。例えばリセット後、図2の一連の処理を何度か行い、全てのアプリケーションから準備完了の通知がある状態となると、準備状態αから作動状態βに遷移し、このときこの状態遷移時に実行すべき各APL内のルーチンを実行する。

【0025】また、各APLは自己の制御対象の装置からの情報を監視し、制御が完了すると制御完了の通知を行う。例えば自動車が駐車状態になった場合には、図2の一連の処理を何度か行っていると全てのAPLが制御完了の通知を行う状態になる。したがって、作動状態βから作動待ち状態γへ遷移し、このときこの状態遷移時に実行すべき各APL内のルーチンを実行する。すなわち、作動状態βは各APLの処理によって人が自動車に対して何らかの操作をしているかあるいは操作を行う可能性があるかと判定されている状態であり、作動待ち状態γはワイヤレスの電波待ちやスマートキーの応答待ちなど、人が自動車に対してアクションを開始しようとするエッジの検出を行うための状態である。

【0026】そして作動待ち状態γでは、次に図2の一連の処理を実行した後、各APLからの制御開始の通知がなければ、ECUの動作を停止するスリープへ遷移させて消費電力を低減させる。一方、ECUは所定の時間（例えば100ms）毎にウェイクアップされ、図2の一連の処理を実行する。このときいずれかのAPLから制御開始の通知があれば作動待ち状態γから作動状態βへ状態を遷移させ、この状態遷移時に実行すべき各APL内のルーチンを実行する。

【0027】このように各APLは自APLによる実行結果をプラットフォームに通知するだけでよく、状態の管理はプラットフォームの制御状態判定処理によって一括して行われる。そのため、各々のAPLで相互に他のAPLの状態（実行結果）を参照して現在の状態を求める必要がない。したがって、従来こうした処理に必要な

った記憶領域やアプリケーションのプログラムは不要となり、こうした処理にかかる処理時間も不要となる。

【0028】さらに状態の遷移時点を各アプリケーション毎に捉えて、その状態遷移時に実行すべき自己のルーチンを各アプリケーションによって実行させる必要もなくなる。すなわち、状態の変化を捉えるためのプログラムや記憶領域を各アプリケーション毎に設ける必要がなくなる。またこうした処理にかかる処理時間も不要となる。

【0029】このように、アプリケーション毎のリソース消費を減らすことができ、アプリケーション毎の設計及び実装にかかるコストを軽減することができる。特にアプリケーションの数が多くなればなるほど効果が大きくなる。

\* くなる。

【図面の簡単な説明】

【図1】 ECUの動作概要を示すフローチャートである。

【図2】 プラットフォーム、アプリケーション、制御状態判定の各プログラムによる処理の関係を示す説明図である。

【図3】 制御状態判定処理による状態遷移を示す説明図である。

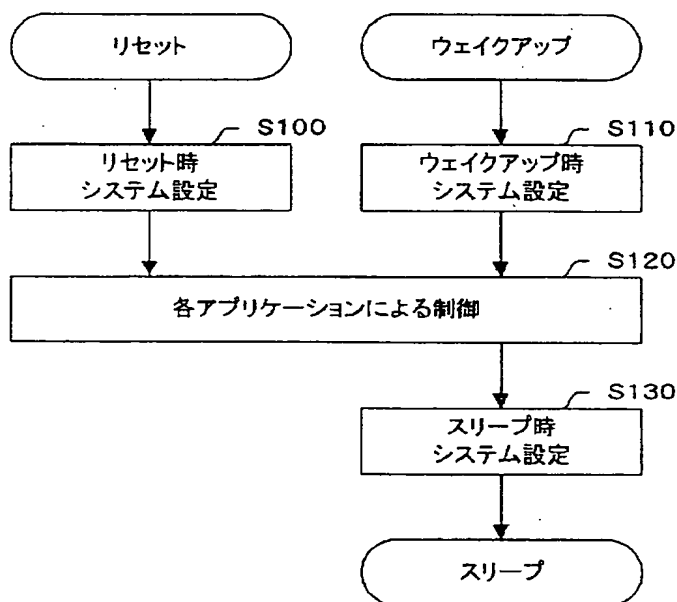
【符号の説明】

$\alpha$  … 準備状態

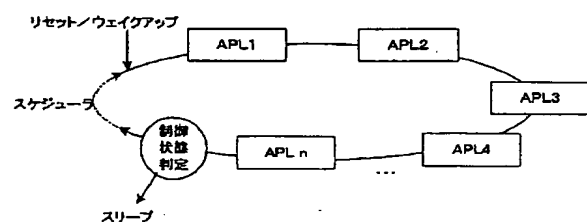
$\beta$  … 作動状態

$\gamma$  … 作動待ち状態

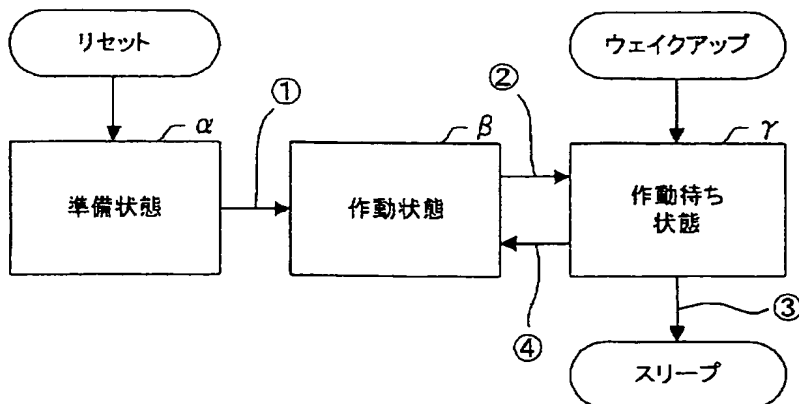
【図1】



【図2】



【図3】



フロントページの続き

(72)発明者 江川 邦隆

Fターム(参考) 5B098 GC02

愛知県刈谷市昭和町1丁目1番地 株式会  
社デンソー内